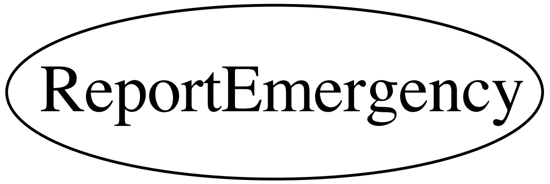
Next goal, after the scenarios are formulated

- Find all the use cases in the scenario that specify all instances of how to report a fire
  - **Example: “Report Emergency“ in the first paragraph of the scenario is a candidate for a use case**
- Describe each of these use cases in more detail
  - **Participating actors**
  - **Describe the entry condition**
  - **Describe the flow of events**
  - **Describe the exit condition**
  - **Describe exceptions**
  - **Describe nonfunctional requirements (constraints, nonfunctional requirements)**

Use Cases

- A use case is a flow of events in the system, including interaction with actors
- It is initiated by an actor
- Each use case has a name
- Each use case has a termination condition
- Graphical Notation: An oval with the name of the use case



- *Use Case Model:* **The set of all use cases specifying the complete functionality of the system**

How to find Use Cases

- Select a narrow vertical slice of the system (i.e. one scenario)
  - **Discuss it in detail with the user to understand the user's preferred style of interaction**
- Select a horizontal slice (i.e. many scenarios) to define the scope of the system.
  - **Discuss the scope with the user**
- Use illustrative prototypes (mock-ups) as visual support
- Find out what the user does
  - **Task observation (Good)**
  - **Questionnaires (Bad)**

Steps in Formulating a Use Case

- First step: name the use case
  - **Use case name: ReportEmergency**
- Second Step: find the actors
  - **Generalize the concrete names (“Bob”) to participating actors (“Field officer”)**
  - **Participating Actors: ReportEmergency**
    - **Field Officer (Initiator) (Bob and Alice in the scenario)**
    - **Dispatcher (John in the scenario)**
- Third step: concentrate on the flow of events
  - **Use informal natural language**

Use Case Example: ReportEmergency

- Use case name: ReportEmergency
- Participating Actors:
  - **Field Officer (Bob and Alice in the Scenario)**
  - **Dispatcher (John in the Scenario)**
- Exceptions:
  - **The FieldOfficer is notified immediately if the connection between terminal and central is lost.**
  - **The Dispatcher is notified immediately if the connection between a FieldOfficer and central is lost.**
- Flow of Events: **on next slide.**
- Special Requirements:
  - **The FieldOfficer's report is acknowledged within 30 seconds. The selected response arrives no later than 30 seconds after it is sent by the Dispatcher.**

Use Case Example: ReportEmergency Flow of Events

1. The **FieldOfficer** activates the “Report Emergency” function of her terminal. FRIEND responds by presenting a form to the officer.
2. The FieldOfficer fills the form, by selecting the emergency level, type, location, and brief description of the situation. The FieldOfficer also describes a response to the emergency situation. Once the form is completed, the FieldOfficer submits the form, and the **Dispatcher** is notified.
3. The Dispatcher creates an Incident in the database by invoking the OpenIncident use case. He selects a response and acknowledges the report.
4. The FieldOfficer receives the acknowledgment and the selected response.

Example of steps in formulating a use case

- Write down the exceptions:
  - **The FieldOfficer is notified immediately if the connection between her terminal and the central is lost.**
  - **The Dispatcher is notified immediately if the connection between any logged in FieldOfficer and the central is lost.**
- Identify and write down any special requirements:
  - **The FieldOfficer's report is acknowledged within 30 seconds.**
  - **The selected response arrives no later than 30 seconds after it is sent by the Dispatcher.**

Another Example: Allocate a Resource

- Actors:
  - ***Field Supervisor:*** **This is the official at the emergency site.**
  - ***Resource Allocator:*** **The Resource Allocator is responsible for the commitment and decommitment of the Resources managed by the FRIEND system.**
  - ***Dispatcher:*** **A Dispatcher enters, updates, and removes Emergency Incidents, Actions, and Requests in the system. The Dispatcher also closes Emergency Incidents.**
  - ***Field Officer:*** **Reports accidents from the Field**

Another Example: Allocate a Resource

- Use case name: AllocateResources
- Participating Actors:

  Field Officer (Bob and Alice in the Scenario)

  Dispatcher (John in the Scenario)

  Resource Allocator and Field Supervisor
- Entry Condition:

  The Resource Allocator has selected an available resource

  The resource is currently not allocated
- Flow of Events:

  1. The Resource Allocator selects an Emergency Incident

  2. The Resource is committed to the Emergency Incident
- Exit Condition:

  The use case terminates when the resource is committed

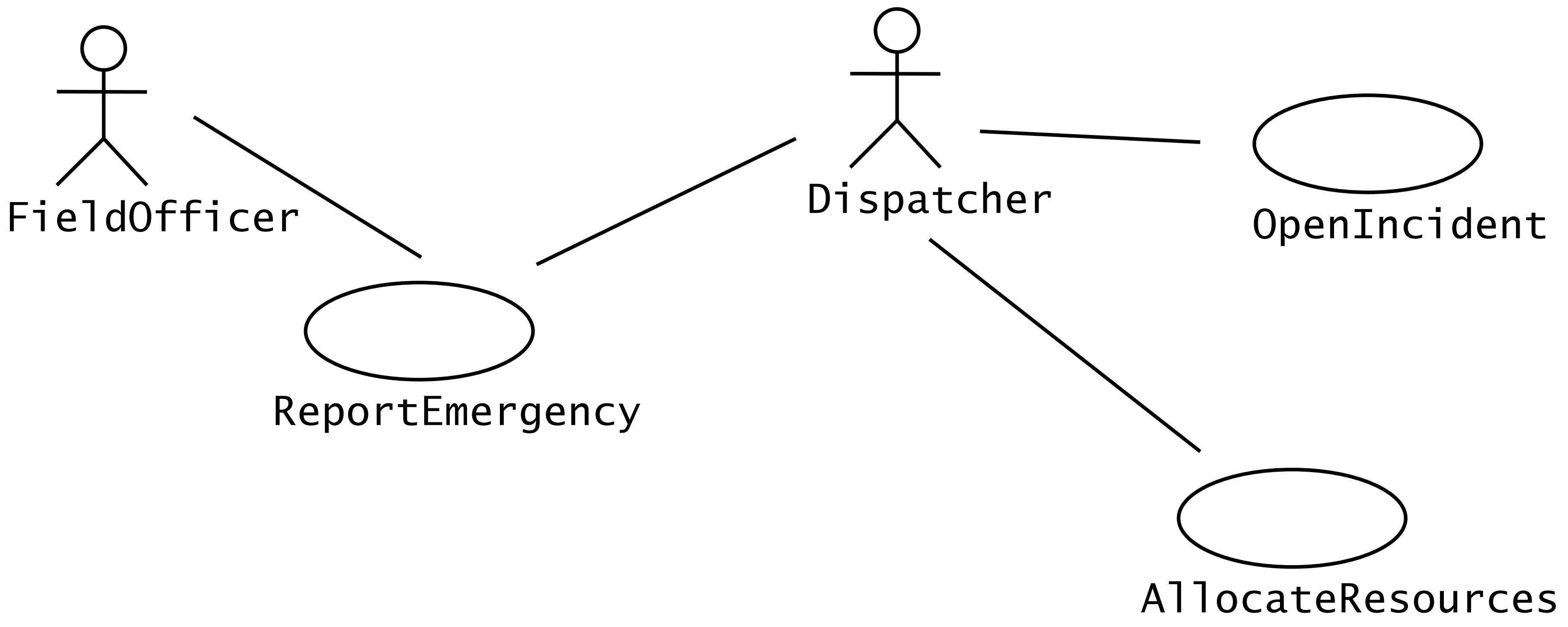
  The selected Resource is unavailable to other Requests.
- Special Requirements:

  The Field Supervisor is responsible for managing Resources

Come specificare un caso d'uso (Sommario)

- Nome del Caso d'uso
- Attori
  - **Descrizione degli attori coinvolti nel caso d'uso**
- Entry condition
  - **Usare una frase come "Questo caso d'uso inizia quando…"**
- Flusso di Eventi
  - **Forma libera, linguaggio naturale informale**
- Exit condition
  - **Iniziare con "Questo caso d'uso termina quando…"**
- Eccezioni
  - **Descrivere cosa accade quando le cose vanno male**
- Requisiti Speciali
  - **Elencare i requisiti non funzionali e i vincoli**

Use Case Model for Incident Management



Another Use Case Example

Actor **Bank Customer**

- Person who owns one or more Accounts in the Bank.

Withdraw Money

- The Bank Customer specifies a Account and provides credentials to the Bank proving that s/he is authorized to access the Bank Account.
- The Bank Customer specifies the amount of money s/he wishes to withdraw.
- The Bank checks if the amount is consistent with the rules of the Bank and the state of the Bank Customer's account. If that is the case, the Bank Customer receives the money in cash.

Use Case Attributes

Use Case **Withdraw Money Using ATM**

Initiatiating actor:

- Bank Customer

Entry conditions:

- Bank Customer has opened a Bank Account with the Bank ***and***
- Bank Customer has received an ATM Card and PIN

Exit conditions:

- Bank Customer has the requested cash ***or***
- Bank Customer receives an explanation from the ATM about why the cash could not be dispensed

Use Case Flow of Events

Actor steps	System steps
1.The Bank Customer inputs the card into the ATM.	
	2.The ATM requests the input of a four-digit PIN.
3. The Bank Customer types in PIN.	
	4. If several accounts are recorded on the card, the ATM offers a choice of the account numbers for selection by the Bank Customer
5. The Bank Customer selects an account.	
	6.If only one account is recorded on the card or after the selection, the ATM requests the amount to be withdrawn.
7. The Bank Customer inputs an amount.	
	8.The ATM outputs the money and a receipt and stops the interaction.

Use Case Exceptions

Actor steps

1. The Bank Customer inputs her card into the ATM.**[Invalid card]**

3. The Bank Customer types in PIN. **[Invalid PIN]**

5. The Bank Customer selects an account .

7. The Bank Customer inputs an amount. **[Amount over limit]**

[Invalid card]

The ATM outputs the card and stops the interaction.

[Invalid PIN]

The ATM announces the failure and offers a 2nd try as well as canceling the whole use case. After 3 failures, it announces the possible retention of the card. After the 4th failure it keeps the card and stops the interaction.

[Amount over limit]

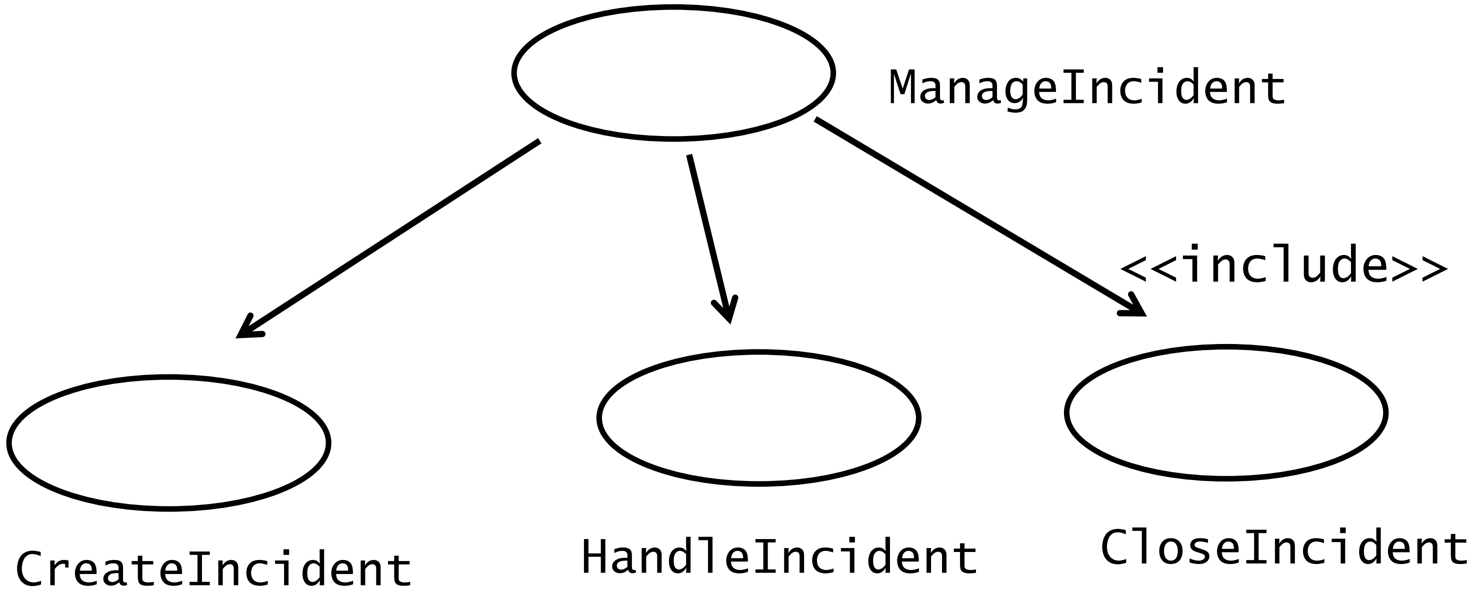
The ATM announces the failure and the available limit and offers a second try as well as canceling the whole use case.

Use Case Associations

- Dependencies between use cases are represented with use case associations
- Associations are used to reduce complexity
  - **Decompose a long use case into shorter ones**
  - **Separate alternate flows of events**
  - **Refine abstract use cases**
- Types of use case associations
  - **Includes**
  - **Extends**
  - **Generalization**

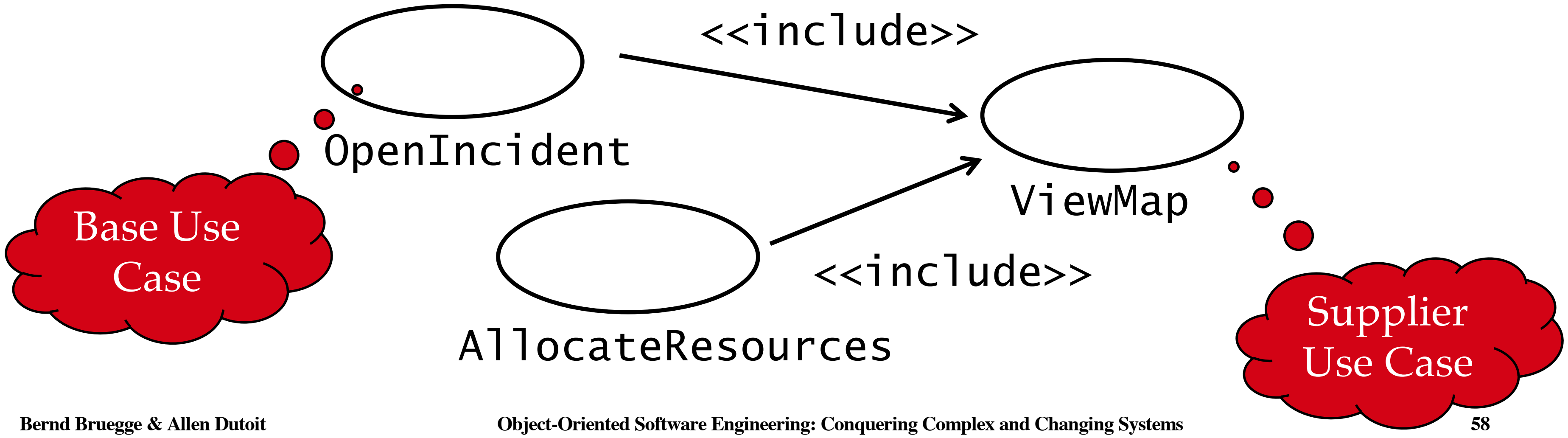
<<include>>: Functional Decomposition

- Problem:
  - **A function in the original problem statement is too complex**
- Solution:
  - **Describe the function as the aggregation of a set of simpler functions. The associated use case is decomposed into shorter use cases**



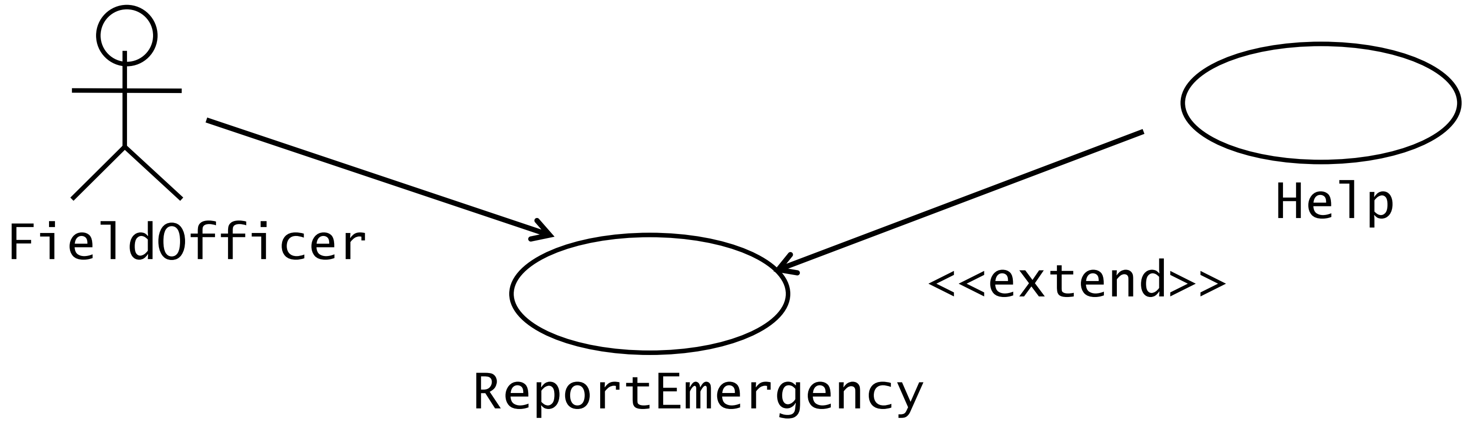
<<include>>: Reuse of Existing Functionality

- Problem: There are overlaps among use cases. How can we *reuse* flows of events instead of duplicating them?
- Solution: The *include relationship* from use case A to use case B indicates that an instance of use case A performs all the behavior described in use case B ("A delegates to B")
- Example: Use case "ViewMap" describes behavior that can be used by use case "OpenIncident" ("ViewMap" is factored out)
- Note: the base case cannot exist alone. It is always called with the supplier case



<<extend>> Association for Use Cases

- Problem: The functionality in the original problem statement needs to be extended.
- Solution: An *extend relationship* from use case A to use case B indicates that use case A is an extension of use case B.
- Example: "ReportEmergency" is complete by itself, but can be extended by use case "Help" for a scenario in which the user requires help

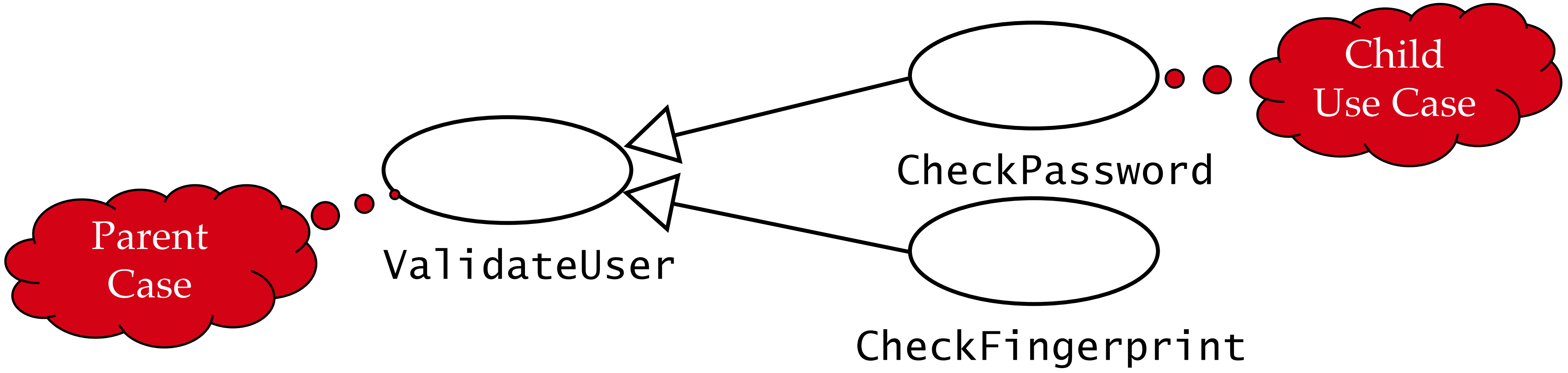


Association for Use Cases

- Extend
  - **Non occorre modificare lo /gli use case che vengono estesi**
  - **L'attivazione può avvenire in un punto qualsiasi del flusso di eventi dello use case base**
  - **Sono spessi situazioni eccezionali (failure, cancel, help...)**
- Include
  - **Ogni use case che include deve specificare dove viene invocato lo use case incluso**
  - **Funzioni comuni a più use case**
- Esempio Fig. 4-14 libro

Generalization relationship in Use Cases

- Problem: You have common behavior among use cases and want to factor this out.
- Solution: The generalization relation among use cases factors out common behavior. The child use cases inherit the behavior and meaning of the parent use case and add or override some behavior.
- Example: "ValidateUser" is responsible for verifying the identity of the user. The customer might require two realizations: "CheckPassword" and "CheckFingerprint"


Child Use Case
CheckPassword
Parent Case
ValidateUser
CheckFingerprint